- Collections
- List(ArrayList)
- 1. Search an Element
- Write a program to:
- Create an ArrayList of integers.
- Ask the user to enter a number.
- Check if the number exists in the list.

```
package Assesement_day8;
import java.util.ArrayList;

import java.util.Scanner;

public class Array {

        public static void main(String[] args) {

                ArrayList<Integer> numbers = new ArrayList<>();

                numbers.add(10);

                numbers.add(20);

                numbers.add(30);

                numbers.add(40);

                numbers.add(50);
                Scanner scanner = new Scanner(System.in);
                System.out.print("Enter a number: ");
                int target = scanner.nextInt();
                scanner.close();
                if (numbers.contains(target)) {
                    System.out.println(target + " exists in the list.");
                } else {
                    System.out.println(target + " does not exist in the list.");
                }
        }
}
```

Output:

```
Enter a number: 30
30 exists in the list.
```

- 3. Remove Specific Element
- Write a program to:
- Create an ArrayList of Strings.

- **Add 5 fruits.**
- **Remove a specific fruit by name.**
- **Display the updated list.**

```
package Assessement_day8;
import java.util.ArrayList;

public class remove_element {

public static void main(String[] args) {
ArrayList<String> list=new ArrayList <>();
list.add("mango");
list.add("apple");
list.add("banana");
list.add("grape");
System.out.println(list);
list.remove("apple");
System.out.println(list);

}

}
```

Output:

```
[mango, apple, banana, grape]
[mango, banana, grape]
```

3. Sort Elements

- Write a program to:
- Create an ArrayList of integers.
- Add at least 7 random numbers.
- Sort the list in ascending order.
- Display the sorted list.

```
package Assesement_day8;
    import java.util.ArrayList;

    import java.util.Collections;

    public class Sort_integer_asc {

            public static void main(String[] args) {

                    ArrayList<Integer> numbers = new ArrayList<>();

                    numbers.add(34);

                    numbers.add(12);

                    numbers.add(56);

                    numbers.add(7);

                    numbers.add(23);

                    numbers.add(90);

                    numbers.add(45);
```

```
                System.out.println("Before sorting: " +
    numbers);


                Collections.sort(numbers);
                System.out.println("After sorting: " +
    numbers);
            }
    }
```

Output:

```
    Before sorting: [34, 12, 56, 7, 23, 90, 45]
    After sorting: [7, 12, 23, 34, 45, 56, 90]
```

- **4. Reverse the ArrayList**
- **Write a program to:**
- **Create an ArrayList of characters.**
- **Add 5 characters.**
- **Reverse the list using Collections.reverse() and display it.**

```
package Assessement_day8;
import java.util.ArrayList;
import java.util.Collections;
public class Reverse_list {
```

```
public static void main(String[] args) {
ArrayList<Character> characters = new ArrayList<>();
characters.add('d');
characters.add('h');
characters.add('l');
characters.add('i');
characters.add('p');

System.out.println("Original list: " + characters);

Collections.reverse(characters);

System.out.println("Reversed list: " + characters);

}

}
```

Output:

```
Original list: [d, h, l, i, p]
Reversed list: [p, i, l, h, d]
```

- 5. Update an Element
- Write a program to:
- Create an ArrayList of subjects.

- Replace one of the subjects (e.g., “Math” to “Statistics”).
- Print the list before and after the update.

```
package Assessement_day8;
import java.util.ArrayList;
public class replace {

public static void main(String[] args) {
ArrayList<String> subjects = new ArrayList<>();
subjects.add("Math");
subjects.add("Science");
subjects.add("Hindi");
subjects.add("English");
subjects.add("Telugu");

System.out.println("Before update: " + subjects);

subjects.set(0, "Statistics");

System.out.println("After update: " + subjects);

}

}
```

Output:

```
Before update: [Math, Science, Hindi, English, Telugu]
After update: [Statistics, Science, Hindi, English, Telugu]
```

- 6. Remove All Elements
- Write a program to:
- Create an ArrayList of integers.
- Add multiple elements.
- Remove all elements using clear() method.
- Display the size of the list.

```
package Assessement_day8;
import java.util.ArrayList;
public class remove_elements {

public static void main(String[] args) {
ArrayList<Integer> numbers = new ArrayList<>();
numbers.add(10);
numbers.add(30);
numbers.add(80);
numbers.add(20);
numbers.add(50);
System.out.println("**********Before clearing**********");
System.out.println( numbers);
System.out.println("Size: " + numbers.size());

numbers.clear();
System.out.println("**********After
clearing**********");
System.out.println(numbers);
System.out.println("Size: " + numbers.size());

}

}
```

Output:

```
**********Before clearing**********
[10, 30, 80, 20, 50]
Size: 5

**********After clearing**********
[]
Size: 0
```

- **List(LinkedList)**
- **1. Create and Display a LinkedList**
- **Write a program to:**
- **Create a LinkedList of Strings.**
- **Add five colors to it.**
- **Display the list using a for-each loop.**

```
package Assesement_day8;
    import java.util.LinkedList;
```

```
public class Linkedlist {
        public static void main(String[] args) {
                LinkedList<String> colors = new
LinkedList<>();
                colors.add("Red");
                colors.add("Green");
                colors.add("Blue");
                colors.add("Yellow");
                colors.add("Purple");
                System.out.println("Colors:");
                for (String color : colors) {
                        System.out.println(color);
                }
        }
}
```

Output:

```
Colors:
Red
Green
Blue
```

Yellow

Purple

- 2. Add Elements at First and Last Position
- Write a program to:
- Create a LinkedList of integers.
- Add elements at the beginning and at the end.
- Display the updated list.

```
package Assesement_day8;
 import java.util.LinkedList;
 public class Add_elements {
 	public static void main(String[] args) {
 		LinkedList<Integer> numbers = new LinkedList<>();
 		numbers.add(10);
 		numbers.add(20);
 		numbers.add(30);

 		System.out.println("Original list: " + numbers);
```

```
                numbers.addFirst(8);
                numbers.addLast(90);

                System.out.println("Updated list: " +
numbers);
        }
}
```

Output:

```
Original list: [10, 20, 30]
Updated list: [8, 10, 20, 30, 90]
```

- **3. Insert Element at Specific Position**
- **Write a program to:**
- **Create a LinkedList of names.**
- **Insert a name at index 2.**
- **Display the list before and after insertion.**

```
package Assessement_day8;
import java.util.LinkedList;
```

```
public class Insert_emnt {
public static void main(String[] args) {
LinkedList<String> names = new LinkedList<>();
names.add("sanjana");
names.add("sri");
names.add("dhana");
names.add("penugonda");

System.out.println("Before insertion: " + names);

names.add(2, "prasanna");

System.out.println("After insertion: " + names);
}

}
```

Output:

```
Before insertion: [sanjana, sri, dhana, penugonda]
After insertion: [sanjana, sri, prasanna, dhana,
penugonda]
```

- 4. Remove Elements
- Write a program to:
- Create a LinkedList of animal names.

- Remove the first and last elements.
- Remove a specific element by value.
- Display the list after each removal.

```
package Assessement_day8;
import java.util.LinkedList;

public class removing_elements {

public static void main(String[] args) {
LinkedList<String> animals = new LinkedList<>();
animals.add("Lion");
animals.add("Tiger");
animals.add("Elephant");
animals.add("dog");
animals.add("Zebra");

System.out.println("Original list: " + animals);

animals.removeFirst();
System.out.println("After removing first element: " +
animals);

animals.removeLast();
System.out.println("After removing last element: " +
animals);

animals.remove("Elephant");
System.out.println("After removing 'Elephant': " +
animals);
}

}
```

Output:

```
Original list: [Lion, Tiger, Elephant, dog, Zebra]
After removing first element: [Tiger, Elephant, dog,
Zebra]
After removing last element: [Tiger, Elephant, dog]
After removing 'Elephant': [Tiger, dog]
```

- **5. Search for an Element**
- **Write a program to:**
- **Create a LinkedList of Strings.**
- **Ask the user for a string to search.**
- **Display if the string is found or not.**

```
package Assessement_day8;
import java.util.LinkedList;

public class Searching_for_element {

public static void main(String[] args) {
LinkedList<String> list = new LinkedList<>();
list.add("Apple");
list.add("Banana");
list.add("Cherry");
list.add("Date");

String searchString = "mango";

if (list.contains(searchString)) {
System.out.println(searchString + " found in the list.");
} else {
System.out.println(searchString + " not found in the
list.");
}

}


}
```

Output:

```
    mango not found in the list.
```

- 6. Convert LinkedList to ArrayList
- Write a program to:
- Create a LinkedList of Strings.
- Convert it into an ArrayList.
- Display both the LinkedList and ArrayList.

```
package Assesement_day8;
 import java.util.ArrayList;
import java.util.LinkedList;
public class Linked_Array {
        public static void main(String[] args) {
                LinkedList<String> linkedList = new LinkedList<>();
                linkedList.add("Apple");
                linkedList.add("Banana");
                linkedList.add("Cherry");

                ArrayList<String> arrayList = new ArrayList<>(linkedList);

                System.out.println("LinkedList: " + linkedList);
                System.out.println("ArrayList: " + arrayList);
        }
}
```

Output:

```
LinkedList: [Apple, Banana, Cherry]
```

```
 ArrayList: [Apple, Banana, Cherry]
```

- **Vector**
- Create a Vector of integers and perform the following operations:
- Add 5 integers to the Vector.
- Insert an element at the 3rd position.
- Remove the 2nd element.
- Display the elements using Enumeration.

```
package Assesement_day8;
import java.util.Enumeration;
import java.util.Vector;
public class Main {
	public static void main(String[] args) {
		Vector<Integer> vector = new Vector<>();
		vector.add(10);
		vector.add(20);
		vector.add(30);
		vector.add(40);
		vector.add(50);
}


        System.out.println("Original Vector: " +
vector);


        vector.add(2, 25);
        System.out.println("After insertion: " + vector);


        vector.remove(1);
        System.out.println("After removal: " + vector);


        Enumeration<Integer> enumeration =
vector.elements();
        System.out.println("Elements using
Enumeration:");
        while (enumeration.hasMoreElements()) {

System.out.println(enumeration.nextElement());
        }
    }
}
```

Output:

```
Original Vector: [10, 20, 30, 40, 50]
After insertion: [10, 20, 25, 30, 40, 50]
After removal: [10, 25, 30, 40, 50]
Elements using Enumeration:
10
25
30
40
50
```

- **Create a Vector of Strings and:**
- **Add at least 4 names.**
- **Check if a specific name exists in the vector.**
- **Replace one name with another.**
- **Clear all elements from the vector.**

```
package Assessement_day8;
import java.util.Vector;
public class Vector_main {

public static void main(String[] args) {
Vector<String> names = new Vector<>();
names.add("Sanjana");
names.add("Dhana");
names.add("Sri");
names.add("Penugonda");

System.out.println("Original Vector: " + names);

if (names.contains("Sri")) {
System.out.println("Emma exists in the vector.");
}

names.set(2, "Prasanna");
System.out.println("After replacement: " + names);

names.removeAllElements();
System.out.println("After clearing: " + names);
}
}
```

Output:

```
Original Vector: [Sanjana, Dhana, Sri, Penugonda]
Emma exists in the vector.
After replacement: [Sanjana, Dhana, Prasanna,
Penugonda]
After clearing: []
```

- **Stack**
- **Create a Stack of integers and:**

- Push 5 elements.
- Pop the top element.
- Peek the current top.
- Check if the stack is empty.
- Reverse a string using Stack:
- Input a string from the user.
- Use a stack to reverse and print the string.
- Use Stack to check for balanced parentheses in an expression.

```
package Assesement_day8;
import java.util.Stack;
public class Stack {
	public static void main(String[] args) {
		Stack<String> stack = new Stack<>();
		stack.push("A");
		stack.push("B");
		stack.push("C");
		System.out.println("Stack: " + stack);
		System.out.println("Popped element: " +
stack.pop());
		System.out.println("Stack after pop: " + stack);
```

```
                System.out.println("Top element: " +
stack.peek());
                String str = "Hello";
                Stack<Character> charStack = new Stack<>();
                for (char c : str.toCharArray()) {
                        charStack.push(c);
                }
                System.out.print("Reversed string: ");
                while (!charStack.isEmpty()) {
                        System.out.print(charStack.pop());
                }
        }
}
```

Output:

```
Stack: [A, B, C]
Popped element: C
Stack after pop: [A, B]
Top element: B
Reversed string: olleH
```

- **HashSet**

- Create a HashSet of Strings:
- Add 5 different city names.
- Try adding a duplicate city and observe the output.
- Iterate using an Iterator and print each city.
- Perform operations:
- Remove an element.
- Check if a city exists.
- Clear the entire HashSet.

```
package Assessement_day8;
import java.util.HashSet;
public class Hashset {

public static void main(String[] args) {
HashSet<String> cities = new HashSet<>();
cities.add("London");
cities.add("Paris");
cities.add("Rome");
cities.add("Tokyo");
cities.add("London");
System.out.println("Cities: " + cities);

cities.remove("Rome");
System.out.println("After removing Rome: " + cities);
System.out.println("Does cities contain Paris? " +
cities.contains("Paris"));

cities.clear();
System.out.println("After clearing: " + cities);
System.out.println("Is cities empty? " + cities.isEmpty());
}

}
```

Output:

```
Cities: [Rome, Tokyo, London, Paris]
After removing Rome: [Tokyo, London, Paris]
Does cities contain Paris? true
After clearing: []
Is cities empty? true
```

- LinkedHashSet
- 1.Create a LinkedHashSet of Integers:
- Add numbers: 10, 5, 20, 15, 5.
- Print the elements and observe the order.

```
package Assesement_day8;
import java.util.LinkedHashSet;

public class Main {

        public static void main(String[] args) {
```

```
        LinkedHashSet<Integer> set = new
LinkedHashSet<>();
        set.add(10);
        set.add(5);
        set.add(20);
        set.add(15);
        set.add(5);
        System.out.println("Elements: " + set);
    }
}
```

Output:

```
Elements: [10, 5, 20, 15]
```

- **Write a program to:**
- **Merge two LinkedHashSets and print the result**

```
package Assesement_day8;
import java.util.LinkedHashSet;
public class Main {
    public static void main(String[] args) {
        LinkedHashSet<String> set1 = new
LinkedHashSet<>();
        set1.add("Apple");
```

```
                set1.add("Banana");
                set1.add("Cherry");
                LinkedHashSet<String> set2 = new
LinkedHashSet<>();
                set2.add("Date");
                set2.add("Elderberry");
                set2.add("Banana");
                set1.addAll(set2);
                System.out.println("Merged Set: " + set1);
        }
}
```

Output:

```
Merged Set: [Apple, Banana, Cherry, Date, Elderberry]
```

- **TreeSet**
- **1. Create a TreeSet of Strings:**
- **Add 5 country names in random order.**
- **Print the sorted list of countries using TreeSet.**

```
package Assesement_day4;
import java.util.TreeSet;
public class TreeSet {
        public static void main(String[] args) {
```

```
        TreeSet<String> countries = new TreeSet<>();
        countries.add("Brazil");
        countries.add("India");
        countries.add("Australia");
        countries.add("China");
        countries.add("Germany");

        System.out.println("Sorted Countries: " + countries);
    }
}
```

Output:

```
Sorted Countries: [Australia, Brazil, China, Germany, India]
```

- **2.Create a TreeSet of Integers:**
- **Add some numbers and print the first and last elements.**
- **Find the elements lower than and higher than a given number using lower() and higher() methods.**

```
package Assesement_day8;
import java.util.TreeSet;
```

```
}

public class Main {
        public static void main(String[] args) {
                TreeSet<Integer> set = new TreeSet<>();
                set.add(10);
                set.add(5);
                set.add(20);
                set.add(15);
                set.add(25);
                System.out.println("TreeSet: " + set);
                        System.out.println("First element: " +
set.first());
                System.out.println("Last element: " +
set.last());
                int num = 15;
                System.out.println("Lower than " + num + ": "
+ set.lower(num));
                System.out.println("Higher than " + num + ": "
+ set.higher(num));
        }
}
```

Output:

```
TreeSet: [5, 10, 15, 20, 25]

First element: 5

Last element: 25

Lower than 15: 10

Higher than 15: 20
```